

HALCON Version 5.2



Getting Started
User's Manual

December 19, 2001

Getting started with HALCON, Version 5.2

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher.

Edition 1	July 1997
Edition 2	November 1997
Edition 3	March 1998
Edition 4	April 1999
Edition 4a	December 2001



Copyright © 1997-2001 MVTec Software GmbH, Munich, Germany

Information concerning HALCON:
<http://www.mvtec.com>

Contents

1	About this Manual	1
1.1	Readers	1
1.2	Organization of this Manual	1
1.3	Notation	2
1.4	Additional Sources of Information	2
1.5	Release Notes	3
2	Getting Started	5
2.1	What is HALCON?	5
2.2	How is HALCON Related to HORUS?	6
2.3	HALCON Installation	6
2.3.1	Windows NT and Windows 2000	7
2.3.2	Unix	11
2.4	HALCON Licensing	11
2.5	HALCON Configuration	13
2.5.1	Windows NT Version	13
2.5.2	Unix Version	13
2.6	A First Look at HDevelop	15
2.6.1	Main window	16
2.6.2	Program Window	17
2.6.3	Operator Window	17
2.6.4	Variable Window	17
2.6.5	Graphics Window	17
2.7	Troubleshooting	17
2.8	Help During a HDevelop Session	18
2.9	Limitations	19
3	HDevelop Example Session	21
3.1	Editing a HDevelop Program	22
3.2	Loading an Image	23
3.3	The Graphics Window	24
3.4	Generating a Region of Interest (ROI)	25
3.5	Finding the Right Operator	27
3.6	Finding the Right Parameter Values	28
3.7	Finding the Board as ROI	29
3.8	Finding Bonding Balls Using Morphology	30
3.9	Working with Control Variables	32

3.10	Finding Bonding Balls Using Pattern Matching	33
3.11	Using Control Constructs	36
4	Philosophy of HALCON	37
4.1	Facts about HALCON	37
4.2	Handling Iconic and Control Data	40
4.2.1	Tuple Processing	40
4.2.2	Image Objects	40
4.2.3	Region Objects	41
4.2.4	XLD Objects	42
	Figures	46
	Index	48

Chapter 1

About this Manual

This manual is a quick guide to HALCON — the software solution for machine vision applications. HALCON is a sophisticated image analysis package suitable for product development, research, and education. It provides operators covering a wide range of applications: Factory automation, quality control, remote sensing and aerial image interpretation, medical image analysis, and surveillance tasks. Rapid prototyping is facilitated by the *Computer Aided Vision Engineering (CAVE)* tool HDevelop, which offers a highly interactive programming environment to design image analysis programs.

This manual provides all necessary information to install and configure HALCON, to understand its basic philosophy, and to get some insight in HDevelop.

1.1 Readers

This manual is intended for all new users of HALCON. It does not assume that you are an expert in image processing. Regardless of your skills, it is quite easy to work with HALCON using HDevelop. Anybody should be able to understand the basic HALCON principles to solve his image analysis problems quickly. Nevertheless, it is helpful to have an idea about the functionality of *graphical user interfaces (GUI)*,¹ and about some basic image processing aspects (see e.g., [BB82][Jai92][Rus92]).

1.2 Organization of this Manual

This manual is divided into the following parts:

- **Getting Started**

This chapter provides a short overview of the HALCON system. It explains the installation and configuration process and it introduces the CAVE tool HDevelop.

¹Consult your platform's documentation for general information.

- **HDevelop Example Session**

This chapter contains a complete HDevelop session showing how to solve a specific image analysis task from the scratch. Every user interaction is explained in detail to introduce the most important aspects of HDevelop's GUI and its functionality.

- **Philosophy of HALCON**

Read this chapter to obtain background information about HALCON and its functionality. Especially, it might be helpful to understand the basic iconic data structures and how to handle these objects.

1.3 Notation

Except of Unix specific sections in this manual file paths and environment variables are printed in the Windows NT convention, e.g.,

`%HALCONROOT%\examples\c_interface\halconuser`

to denote the subdirectory `halconuser` containing an example package within the HALCON base directory referenced by the environment variable `HALCONROOT`. The same expression in Unix convention would look like

`$HALCONROOT/examples/c_interface/halconuser`

1.4 Additional Sources of Information

For further information you may consult the following manuals:

- **HDevelop User's Manual**

An introduction to the graphical development environment of the HALCON system.

- **Framegrabber Integration Programmer's Manual**

A guide on how to integrate a new framegrabber in the HALCON system. Note that in some cases you might define new operators (using the C-Interface) instead of using the standard HALCON framegrabber interface in order to exploit specific hardware features of a framegrabber board.

- **HALCON/HDevelop**

The reference manual for all HALCON operators (HDevelop version).

- **HALCON/C++ User's Manual**

How to use the HALCON library in your C++ programs.

- **HALCON/C++**

The reference manual for all HALCON operators (C++ version).

- **HALCON/C User's Manual**
How to use the HALCON library in your C programs.
- **HALCON/C**
The reference manual for all HALCON operators (C version).
- **HALCON/COM User's Manual**
How to use the HALCON library in your COM programs, e.g., in Visual Basic.
- **HALCON/COM**
The reference manual for all HALCON operators (COM version).
- **HALCON C-Interface Programmer's Manual**
How to extend the HALCON system with your own operators.

All these manuals are available as Acrobat Reader (pdf) documents. The reference manuals are available as HTML documents as well. For the latest version of the manuals please check

<http://www.mvtec.com>

1.5 Release Notes

Please note the latest updates of this manual:

- **Edition 4a, HALCON 5.2.4 (December 2001)**
The manual has been adapted to the currently supported architectures.
- **4th Edition, HALCON 5.2 (April 1999)**
The new Apply button is used in the example session. The extended menu and tool bars of HDevelop are described in the introduction. The installation process is described in more detail for Unix systems.
- **3rd Edition, HALCON 5.1 (March 1998)**
The chapter "About This Manual" has been introduced. A complete revision of the manual has been done, e.g., concerning the installation or the licensing section in chapter "Getting Started". The example session has not changed much. The major difference in handling HDevelop in this basic example is that when changing the visualization parameters the iconic object has to be displayed *before* the parameters are changed because the last iconic object is automatically re-displayed.
- **2nd Edition (November 1997)**
The visualization capabilities of HDevelop have been greatly enhanced. Zooming into an image can now be handled very conveniently, for example.

Chapter 2

Getting Started

This chapter provides a short overview of the HALCON system. It explains the installation and configuration process and it introduces the CAVE tool HDevelop. The following aspects are discussed:

- What is HALCON?
- How is HALCON related to HORUS?
- HALCON installation.
- HALCON configuration.
- The main HDevelop windows and their functionality.
- Troubleshooting.
- System help.
- System limitations.

2.1 What is HALCON?

HALCON is a sophisticated image analysis package designed for product development, research, and education. It provides a unique library with more than 850 operators covering a wide range of image analysis tasks, data visualization, and a comfortable debugging mechanism. HALCON operators can be used within C and C++ programs on Windows NT and many Unix platforms. In addition, under Windows NT, HALCON has a COM interface. This allows you to use HALCON operators from Visual Basic, for example. Furthermore, it is very easy to integrate your own specific operators into the system.

The HALCON data structures for iconic data provide a very convenient handling of images, regions, and contours. Optimized algorithms allow high performance even on standard hardware.

The HALCON system also includes the *Computer Aided Vision Engineering (CAVE)* tool HDevelop to speed up the design of image analysis programs. A smart online help based on an *operator knowledge base* provides information about each HALCON operator, e.g., reasonable successors, possible alternatives, cross references, etc. See Fig. 2.1 for an overview of the HALCON architecture.

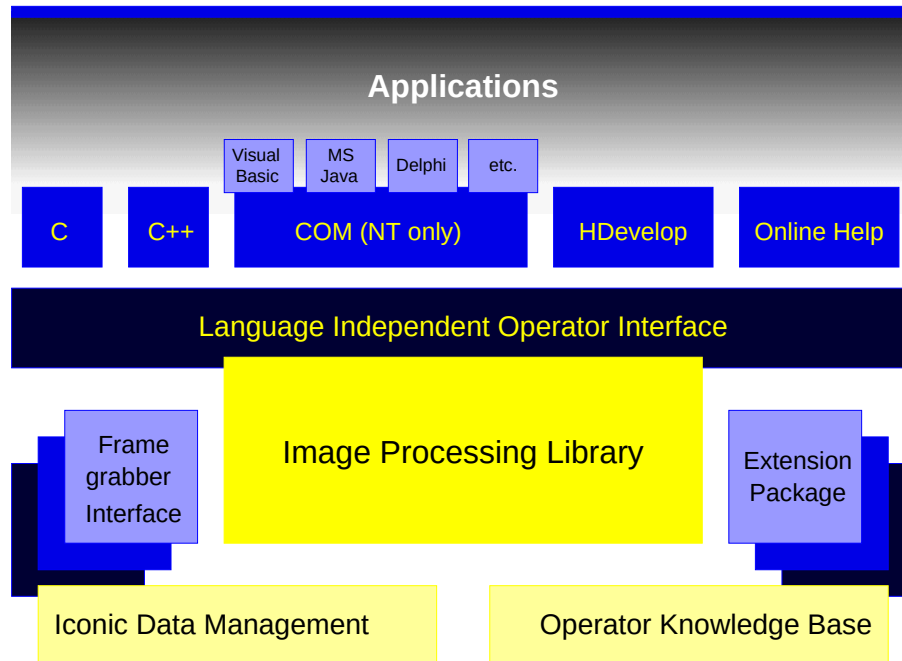


Figure 2.1: The HALCON architecture.

2.2 How is HALCON Related to HORUS?

HALCON is the commercial successor of the image analysis system HORUS,¹ which was developed at the *Technische Universität München* from 1988 to 1996 for Unix systems only. Although MVTec took over the basic concepts of the system, all the major components have been revised. Furthermore, HALCON provides many new features and operators compared to the last HORUS version 4.11. Existing HORUS applications can be adapted to the HALCON system, but since the names and interfaces of many operators have been modified some amount of programming is necessary to do that.

2.3 HALCON Installation

See the file `release_notes.html` within the HALCON package for the latest information. Basically, most of the work is done by the installation scripts provided.

¹The name was changed mainly due to international trademark considerations.

2.3.1 Windows NT and Windows 2000

To install HALCON on Windows NT or Windows 2000 systems, simply insert the CD. This should automatically start the setup program. If the setup program doesn't start automatically, execute the `Setup.exe` program located in the directories `nt-intel` (Intel) of the CD.² The installation wizard allows to specify whether you would like to install

- the full HALCON package (Compact, Custom, Typical),
- the runtime environment only (Runtime), or
- the demo version `hdevelop_demo.exe` (Demo).

This decision influences the license key you will need to run the installed software. For the runtime environment a special *runtime* license is sufficient. The demo version does not need any license key at all. The licensing of a full HALCON package and the runtime version is described in section 2.4.

Furthermore, you can choose between an installation for *node locked* or *floating* licenses. Again, please refer to section 2.4 for more details.

Prior to installing a new HALCON version you should uninstall the old version. This is especially of importance to clean up the registry. Please use either

- the option that the HALCON setup program will show you³ if you have old versions installed (this is the preferred way since it avoids the problem with the path variable described below),
- the system control panel Add/Remove Programs, or
- Start->Programs->Halcon->Uninstall Halcon⁴

to remove the HALCON package. Typically, this is all you have to do. However, please refer to page 8 for trouble shooting if you encounter any problems. Note that an uninstallation will also remove the example images and the documentation. You might want to copy these directories to another directory in order to prevent this. However, we do not recommend this strategy. In most cases there will be both additional images and a revised documentation included in a new HALCON release. Thus, these old directories will be out of date anyway.

Please note that the uninstallation of HALCON versions 5.0 and 5.1 will delete the environment variable `PATH`. This is due to an error in the third-party installation software that is used. If you have installed other software packages that have modified this variable please preserve the old settings prior to the uninstallation: Just use the system control panel System (entry Environment) to rename the variable e.g., to `PATH_`. After the uninstallation rename it again (back to `PATH`) and remove the HALCON specific directories within this variable manually.

²This section describes the installation from CD. The WWW installation is basically the same. However, the installed file groups have been adapted to limit the bandwidth needed for the download.

³For HALCON versions 5.2 and newer only.

⁴For existing HALCON installations newer than version 5.1 only.

If you uninstall a HALCON version 5.0 or 5.1 with floating licenses the FLEXlm license manager service occasionally is not removed properly. To make sure that the service is removed, please start a DOS command prompt prior to the uninstallation and type

```
"%HALCONROOT%\FLEXlm\i486-nt4\installs" -r -n "Halcon Licenses"
```

Please see also the section troubleshooting for the uninstallation on page 8.

Troubleshooting for the Installation

On some systems, you may get a warning message that the HALCON/COM interface library `halconx.dll` failed to self-register. If this happens, the most likely cause for this is that the Microsoft library `at1.dll` is missing on your system. It should reside in the directory `%SystemRoot%\system32`, e.g., `C:\WINNT\system32`. In this case, you must copy this DLL to the directory `%SystemRoot%\system32` manually.⁵ The DLL is located on the CD in the directory `misc\i486-nt4`. After this, you must register the library `at1.dll`. To do so, open a command prompt, change into the directory `misc\i486-nt4` on the CD, e.g., type `E:`, followed by `cd misc\i486-nt4`, and execute the supplied program `reg_halconx` as follows: `reg_halconx at1.dll`. After this, the HALCON/COM library can be registered by executing `reg_halconx` without arguments.

Occasionally, the FLEXlm license manager service needed for floating licenses is not installed properly by the installation wizard. Sometimes you may decide to use floating license after you have installed HALCON. In both cases, you can install and uninstall the license manager manually. In a DOS command prompt, type⁶:

```
"%HALCONROOT%\FLEXlm\i486-nt4\installs" -n "Halcon Licenses"
-c "%HALCONROOT%\license\license.dat" -l "%HALCONROOT%\license\license.log"
-e "%HALCONROOT%\FLEXlm\i486-nt4\lmgrd.exe"
```

where `%HALCONROOT%` is the HALCON base directory you have chosen during the installation. You can check the success of this command using the system control panel Services. If a new service called `Halcon Licenses` appears in the list, you're done.

With the same technique, you can remove the license manager again:

```
"%HALCONROOT%\FLEXlm\i486-nt4\installs" -r -n "Halcon Licenses"
```

Troubleshooting for the Uninstallation

If you had to register the HALCON/COM DLL manually as described above, you must unregister it manually before you can uninstall HALCON. To do so, open a command prompt, change into the directory `misc\i486-nt4` on the CD, e.g., type `E:`, followed by `cd misc\i486-nt4`, and execute the supplied program `reg_halconx` as follows: `reg_halconx /u`.

⁵Unfortunately, a limitation of the third-party installation software prevents us from installing the library from the HALCON setup program.

⁶The following is *one* long command line.

As mentioned above, if you uninstall a HALCON 5.0 or 5.1 version with floating licenses the FLEXlm license manager service occasionally is not removed properly. To make sure that the service is removed, please start a DOS command prompt prior to the uninstallation and type

```
"%HALCONROOT%\FLEXlm\i486-nt4\installs" -r -n "Halcon Licenses"
```

Note that the DOS command prompt cannot handle paths including blanks properly. Thus, if you have installed HALCON to a directory like

```
C:\Program Files\MVTec\Halcon
```

you will have to type either

```
C:\Progra~1\MVTec\Halcon\FLEXlm\i486-nt4\installs -r -n "Halcon Licenses"
```

or quote the command as suggested here

```
"C:\Program Files\MVTec\Halcon\FLEXlm\i486-nt4\installs"  
-r -n "Halcon Licenses"
```

Alternatively, you can also change the directory via

```
cd %HALCONROOT%\FLEXlm\i486-nt4
```

which is possible even without quotes and type

```
installs -r -n "Halcon Licenses"
```

afterwards.

Occasionally, a message like

“Internal Error, unable to load or call external DLL. Please contact your vendor for more information.”

is printed when the uninstaller is invoked and the uninstall process is terminated. This problem is most likely to appear when a new HALCON version was installed over an existing one without completely removing the old files first.

Other possible reasons might be that the whole HALCON directory was moved to another position on the hard disk, or the environment variable %HALCONROOT% was changed manually. You have to follow the following steps to recover from the error:

1. Check if the environment variable %HALCONROOT% matches the location of your HALCON directory. You can check this via the system control panel System (look for Environment) or in a DOS command prompt via `echo %HALCONROOT%`. If the variable contains nothing or the files are in some other location, you have to set HALCONROOT manually via the System control panel.
2. Check if these two files are present:
 - In your %HALCONROOT% directory: `Uninst.isu`

- In the directory %HALCONROOT%\FLEXlm\i486-nt4: HalconUninst.dll⁷

Note that by default the Windows NT Explorer doesn't show DLL files, unless you explicitly tell it to do so. Uninst.isu has been created during the installation. If this file has been removed, an automatic uninstallation is not possible. HalconUninst.dll is a HALCON specific DLL for the uninstallation process. If this file has been removed you can get a copy of the file from the directory FLEXlm\i486-nt4 on the HALCON CD.

3. Check, if the registry entry for the uninstaller is set properly: Start regedit.exe (for example from a DOS command prompt), and go to

```
HKEY_LOCAL_MACHINE -> SOFTWARE -> Microsoft -> Windows ->
CurrentVersion -> Uninstall -> Halcon <Version-Number>,
```

where <Version-Number> is the version number of the HALCON version you want to uninstall or is not present. There, you should find a key named UninstallString. The value of this string should be something⁸ like this:

```
C:\WINNT\IsUninst.exe -f"C:\Program Files\MVTec\Halcon\Uninst.isu"
-c"C:\Program Files\MVTec\Halcon\FLEXlm\i486-nt4\HalconUninst.dll"
```

Make sure that the path C:\Program Files\MVTec\Halcon (or equivalent) in the above example points to the %HALCONROOT% directory. **Note that filenames that contain blanks have to be quoted with "path"**. This is true in the above example since "Program Files" does contain a blank. Older versions of the HALCON setup program failed to do so. If you encounter unquoted path names containing blanks, please insert the quotation marks yourself.

4. Close the registry editor and try to run the uninstaller again.

If the automatical uninstallation fails for whatever reason, proceed as follows:

1. For floating licenses only: Uninstall the license manager service by executing

```
"%HALCONROOT%\FLEXlm\i486-nt4\installs" -r -n "Halcon Licenses"
```

2. Start regedit and delete the keys

```
HKEY_LOCAL_MACHINE -> SOFTWARE -> Microsoft -> Windows ->
CurrentVersion -> Uninstall -> Halcon
```

and

```
HKEY_LOCAL_MACHINE -> SOFTWARE -> MVTec
```

3. Using, e.g., the NT Explorer, delete the directory

```
%SystemRoot%\Profiles\All Users\Start Menu\Programs\Halcon
```

Note that %SystemRoot% usually is C:\Winnt.

4. Finally, delete the HALCON base directory %HALCONROOT%.

⁷For HALCON 5.0 and 5.1, this DLL was called CSUninst.dll.

⁸The option -c... is only included if you have chosen an installation with floating licenses.

2.3.2 Unix

To install HALCON on Unix systems, mount the CD, e.g., on the directory /cdrom. On most systems, you have to have root rights to do so. Note that on some systems, most notably Linux, the default mount entry, usually in /etc/fstab, does not allow user to execute programs on a CD. In such cases, you have to mount the CD explicitly with a command similar to the following:

```
mount /dev/cdrom /cdrom
```

To do the installation, you must⁹ run the install-unix shell script located in the top-level directory of the CD. This script will ask you in which directory you want to install HALCON and about whether to install various components of HALCON. Since all directories and files are read-only on the CD, you may get some warnings about changed permissions if you run the installation script as an ordinary user (as root, no warnings of this kind should appear). Furthermore, on some systems you may get complaints about the fact that the CD contains files with file names longer than 32 characters.

Prior to installing a new HALCON version you should uninstall any old HALCON version. Just remove the contents of the HALCON base directory \$HALCONROOT and all its subdirectories, e.g., by executing

```
rm -rf $HALCONROOT
```

After the installation, several environment variables need to be set in order for HALCON to work. See section 2.5.2 for details.

2.4 HALCON Licensing

HALCON checks whether the product is licensed to the user with the help of the FLEXlm license manager¹⁰ package from GLOBEtrotter. In this section a brief summary of the default procedure to install a license will be given.

There are two possible licensing schemes:

- *Nodelocked* licenses are only valid on one specific host (or for a specific hardware dongle).
- *Floating* licenses are valid on arbitrary hosts within a local network. A license server checks the number of currently active HALCON applications. A new application can be started only if the overall number of licenses is not exceeded.

⁹Well, you should, really, because the filenames are stored as lowercase on the CD only, due to the fact that there is no CD format that works for long file names on both Windows and Unix. The installation script fixes the file names to the correct case.

¹⁰The full documentation of FLEXlm can be obtained from the FLEXlm end user manual available at <http://www.globetrotter.com/manual.htm>.

Note that there are *runtime* licenses available as well. From a user's viewpoint, they are treated exactly the same as the ordinary *development* licenses.

In order to obtain a license for HALCON from MVTec or your local vendor you have to perform the following steps:

- Select your license server node or the computer of your choice for a nodelocked license. License server nodes are machines on a network which keep track of how many licenses of HALCON are currently in use. They need not be a “server” in the sense that they have to be a “big” server machine or have a special kind of operating system, e.g., Windows NT Server, installed. If you want to run HALCON on a stand-alone computer, the license server must be this computer. If you run HALCON on a network any computer on the network can serve as a license server.
- Get the hostid of the selected host. This can be done by running the shipped program `lmhostid` from the shell (Unix) or the shipped program `lmutil` with parameter `lmhostid` from the DOS¹¹ command prompt (Windows NT). An example output of `lmhostid` might look like this on a Windows NT system:

```
> lmutil lmhostid
lmhostid - Copyright (C) 1989-1997 Globetrotter Software, Inc.
The FLEXlm host ID of this machine is "6909b8b2"
```

You must supply your local vendor with the hostid thus obtained (in this example, the string 6909b8b2) and the hostname of this machine in your network (`halconserver`, for example). If the program returns `ffffffff`, this usually indicates that you do not have a network board or, on Windows NT systems, that you do not have the NetBEUI and TCP/IP protocols installed for the network board. If this is the case, please install these two protocols and repeat the above steps.

If you have a Windows NT or Windows 2000 system and no network board installed, or if your computer is not physically connected to a network, you can also use a hardware dongle for licensing. In this case, please send the number printed on the dongle, e.g., `FLEXID=7-b285cb1b`, to your local vendor or MVTec.

- Your local vendor will then supply you with a license file, which you will have to install on your system. Normally, this file must be put into the directory `%HALCONROOT%\license` and be called `license.dat` (see the FLEXlm end user manual on how to configure this location; see the next section for an explanation of the environment variable `HALCONROOT`). A license file typically looks like this:

```
SERVER halconserver 6909b8b2 744
DAEMON mvtecd C:\Progra~1\MVTec\Halcon\FLEXlm\i486-nt4\mvtecd
FEATURE MVTec_Halcon mvtecd 5.2 01-jan-0000 2 4C72DB2EC0F315624FAE \
    VENDOR_STRING=16383 ck=97
FEATURE MVTec_HDevelop mvtecd 5.2 01-jan-0000 2 0C52EBFE396EC7015CD5 \
    VENDOR_STRING=16383 ck=43
```

¹¹Do not start the program from the Explorer. You *must* use a DOS command prompt.

The lines `SERVER` and `DAEMON` are included for floating licenses only. The `SERVER` line tells FLEXlm where to locate the server. You may customize the hostname of the server on the `SERVER` line (the second entry), in case you decide to change the hostname of that computer in your network. You may also customize the port number on the `SERVER` line (the last entry). Finally, depending on where you have installed HALCON, you may have to customize the path to the MVTec daemon `mvtecd`. Normally, this will be `%HALCONROOT%\FLEXlm\%ARCHITECTURE%\mvtecd`. However, since you cannot use environment variables in the license file, you must supply this path manually. Paths including blanks are not handled properly under Windows: Thus, if you have installed HALCON to a directory like `C:\Program Files\MVTec\Halcon` you will have to use the short path `C:\Progra~1\MVTec\Halcon` instead.

- Finally, for floating licenses make sure that the license manager daemon `lmgrd` is started at boot time. On Unix systems this can be done by starting `lmgrd` from the appropriate startup file (usually called `/etc/rc.boot`, `/etc/rc.local`, or `/etc/localrc`). Add the following lines to this file:

```
export HALCONROOT=/usr/local/halcon
/etc/lmgrd -c $HALCONROOT/license/license.dat > /etc/license.log 2>&1 &
```

On Windows NT and Windows 2000 systems the setup program installs¹² the license manager by default as the system service ‘Halcon Licenses’, which will be started automatically at boot time.

2.5 HALCON Configuration

2.5.1 Windows NT Version

All of the work is done by the installation program `Setup.exe` provided. The environment variables `HALCONROOT`, `HALCONIMAGES` and `HALCONEXTENSIONS` are basically the same as in the Unix Version; see section 2.5.2 for details. Please look for the file `release_notes.html` within the HALCON package for the latest information.

2.5.2 Unix Version

The HALCON library and the interactive tool `HDevelop` use some environment variables, which are going to be described in the following. It is reasonable to include them in a *login script* or a *shell resource script* (like `.cshrc`).

- `HALCONROOT`
This is the most important environment variable. It designates the directory, where HALCON is installed. Typical paths are, for example, `/usr/local/halcon` or `/usr/halcon`. According to this variable the system switches to subdirectories, which are important for running HALCON. Some of them are:

¹²See also page 8 for troubleshooting.

- \$HALCONROOT/help
The operator data base is situated in this directory. It is accessible by all HALCON programs to obtain information about HALCON operators.
 - \$HALCONROOT/doc/html/reference/hdevelop
HDevelop uses this directory for online help which can be displayed by a suitable HTML browser like Netscape.
 - \$HALCONROOT/lut
User defined look-up tables are situated in this directory.
 - \$HALCONROOT/ocr
This directory includes trained fonts.
 - \$HALCONROOT/license
This directory contains the license keys necessary for using HALCON.
 - \$HALCONROOT/images
If the variable HALCONIMAGES (see below) is not set the system looks for image files in this directory.
- HALCONIMAGES
To search for image files specified by a relative path, the system uses this environment variable. As a rule it contains several directory names separated by colons on Unix systems and semicolons on Windows NT systems.
 - ARCHITECTURE
Executable HALCON programs reside in \$HALCONROOT/bin/\$ARCHITECTURE. To indicate shared libraries the directory \$HALCONROOT/lib/\$ARCHITECTURE is used. ARCHITECTURE designates the used platform by an abbreviation (e.g., i586-unknown-linux2.2.0; syntax: *processor-hardware_vendor-operating_system*) or does not have any content (i.e., it is empty or undefined), if the installation for HALCON was made for one architecture only. It is useful to include the path \$HALCONROOT/bin/\$ARCHITECTURE or \$HALCONROOT/bin in the path (environment variable PATH) of a shell script.
 - LD_LIBRARY_PATH
The HALCON library path should be included in the system variable LD_LIBRARY_PATH to access shared libraries.
 - HALCONSPY
If this environment variable is defined (regardless of the value) before you start a HALCON program the HALCON spy tool is activated. This corresponds to execute the HALCON operator `set_spy("mode", "on")` in a HALCON program. By this it is possible to monitor an already linked HALCON program during runtime without modifications.

Additionally you may specify any monitoring mode for the HALCON debugging tool. For this you choose a parameter value (e.g. 'operator', 'input_control', 'output_control', etc.) and activate it. This is done by adding these parameters to the environment variable HALCONSPY:

```
setenv HALCONSPY operator=on:input_control=on
```

For further information concerning `set_spy` please refer to the HALCON reference manual.

- **HALCONEXTENSIONS**

This is a list of directories in which user-defined extension operators (so-called packages) are kept. Each package consists of a number of operators linked into a shared library, plus the additional operator documentation in help-files and HTML files. See the C-Interface Programmer's Manual for details on user packages.

- **DISPLAY**

The system uses this environment variable to open windows. It is used in the same way as for other X applications.

- **HOME**

If you start HDevelop the system is looking for a startup file in the home directory (see user manual for HDevelop). The corresponding directory is accessed via the system variable HOME.

Typically the definition of these environment variables belongs to a start shell script like `.cshrc` or `.profile`).

Examples for entries in <code>.cshrc</code>	
<code>setenv HALCONROOT</code>	<code>/usr/local/halcon</code>
<code>setenv ARCHITECTURE</code>	<code>i586-unknown-linux2.2.0</code>
<code>setenv PATH</code>	<code>...:\$HALCONROOT/bin/\$ARCHITECTURE/:...</code>
<code>setenv LD_LIBRARY_PATH</code>	<code>...:\$HALCONROOT/lib/\$ARCHITECTURE/:...</code>
<code>setenv HALCONIMAGES</code>	<code>\$HALCONROOT/images</code>

2.6 A First Look at HDevelop

HDevelop is the CAVE tool of the HALCON system, providing an interactive graphical development environment for image analysis programs. It offers a very easy access to the HALCON functionality. Having started HDevelop, you will see the following windows on your screen:

- *A main interaction window* which includes the HDevelop controls in a menu and a tool bar,
- *a program window,*
- *an operator window,*
- *a variable window, and*
- *a graphics window.*

They comprise the HDevelop *graphical user interface*. Thus it is very important for you to get accustomed with these units. Reading this manual carefully you will not have any problems using HDevelop to solve your image processing tasks.

In the following sections the HDevelop windows will be explained briefly. For more details see Chapter 3 and the HDevelop User's Manual.

2.6.1 Main window

The *main window* is the central window and the starting point for your work. It comprises the creation of a program (operator selection, editing, etc.) and the session management (loading, saving, etc.). This main window is identified by the title 'HDevelop' in the window's title bar. It also contains a *menu bar* and a *tool bar*.

- *Main window menus*

In these menus you will find the complete HDevelop functionality, which is summarized in the following table. For more detailed information see the HDevelop User's Manual.

Menu	Contains Commands for:
File	Loading and saving of HDevelop programs and exiting HDevelop
Edit	Editing a HDevelop program
Execute	Running a HDevelop program
Visualization	Customizing the graphics windows and handling their output behaviour
Operators	Submenus containing programming constructs, HDevelop operators, and all HALCON operators
Suggestions	Support choosing an operator
Window	Window management
Help	Opening online documentation and guides

- *Main window tool bar*

This tool bar offers shortcuts for frequently used commands. Furthermore, it provides the means to control the execution of an HDevelop program.

Button	Performs this Operation:
New	Deletes the current program in the program window
Open	Opens a new HDevelop program
Save	Saves a HDevelop program
Cut	Deletes program lines that have been selected in the program window
Copy	Copies program lines that have been selected in the program window
Paste	Inserts copied lines in the program window
Run	Executes the HDevelop program
Step	Executes the next command of the HDevelop program
Stop	Stops the running HDevelop program
Activate	Activates the lines selected in the program, i.e., makes them executable
Deactivate	Deactivates the lines selected in the program
Reset	Resets the program into its initial state and clears all variables
Set parameters	Opens the visualization parameter dialog window
Pixel info	Opens the pixel info window
Zooming	Opens the online zooming window
Gray histogram info	Opens the interactive gray histogram inspection window
Region info	Opens the interactive region feature inspection window

2.6.2 Program Window

This window is used to display a HDevelop program. This might be an entire program, which has been loaded, or single operators only. It has two parts: a left area containing marks which control the running of a HDevelop program. In the initial HDevelop state you will see the *program counter (PC)*, represented by a green arrow, and the *insertion cursor*. Additionally you may set a *break point* during program debugging (see the user HDevelop User's Manual for a detailed explanation). The right area contains the program code. It is empty in the initial state.

2.6.3 Operator Window

This window will show you important information about a chosen operator. It indicates all parameter names, their types, and their values. It supports you in specifying a parameter value by displaying a default value and alternative values. Additionally, you may get online help for each operator.

Another window element is the text field. If you are not quite sure about an operator name you may specify a substring in this text field. In this case, the system is going to search for all operators in its operator base which include this substring. They are displayed in a *combo box*, where you may choose the appropriate operator.

2.6.4 Variable Window

This is the control unit for iconic and/or control variables generated during the execution of the program. Double-click on an iconic variable to display the contained objects. For a closer look on control variables simply double-click on them.

2.6.5 Graphics Window

This window is used to display iconic variables. The visualization modes can be adapted to your needs with the functions provided in the Visualization menu. You may open an arbitrary number of graphics windows. Each window can be activated interactively or accessed using its specific window ID.

2.7 Troubleshooting

This section explains some typical errors while starting HDevelop and their reasons.

- **No license found**

This error message might have several reasons:

- The file `$HALCONROOT/license/license.dat` is missing and/or not readable.

- Your license is not valid on this machine.
- There are too many licenses active (in case of floating licenses).
- **Lost connection to license server**
Verify if the license server is running. You may also check whether your machine is properly connected with the server. For this you may need to contact your system administrator.
- **No license for this operator**
The operator which you try to execute belongs to a HALCON module which is not licensed. Obtain a new license including this module.
- **Wrong architecture**
This error indicates that you are using an architecture which is not compatible with your HALCON version. Check the ARCHITECTURE variable or try to install the proper HALCON version.
- **hdevelop: Command not found**
Check your system environment variable PATH.
- **lib* : can't open file**
Check whether the system variable LD_LIBRARY_PATH is set correctly (see 2.5.2).
- **No help files for package <package-name> in directory <directory>**
Possible reasons for this error message are:
 - No files \$HALCONROOT/help/* (if the package name is “system”) or no help files in one of the user packages.
 - Wrong HALCONROOT (if the package name is “system”)
 - Check the file protection. Probably HDevelop cannot access important files.
- **Help file for package <package-name> is corrupt**
Possible reasons for this error message are:
 - Inconsistent version of \$HALCONROOT/help/* (if the package name is “system”) or of the help files of one of the user packages.
 - Wrong HALCONROOT (if the package name is “system”)
- **Can't open display**
If you see an error message like this you may have a wrong system variable DISPLAY and/or your program is not allowed to open a window by the specified X-server.

2.8 Help During a HDevelop Session

HDevelop offers you a broad range of support while interacting with HDevelop and while processing image analysis tasks.

- *System suggestions*
Based on the *operator knowledge base* the system offers specific advice to choose an appropriate operator. Thus there are suggestions for reasonable *predecessors* and *successors*. If you are searching for different ways of solving an image analysis problem you may have a look at *alternatives* or *references* of an operator. If you only have some keywords or thematic aspects in mind which an operator has to fulfill, but you cannot remember the operator's name, you have the possibility to obtain an operator by specifying one or more *keywords*.
- *Online help on operators*
HALCON offers a detailed *online help* for each HDevelop operator. These manual pages are highly structured, thus allowing you to get an understanding of an operator easily. To get a survey of the operator's relations, *cross references* to other operators are shown. To use this help you only need an appropriate HTML browser.
- *Debugging help*
You may choose between different *modes of running* a program. You have the possibilities to perform single steps of a program, to set breakpoints or to run a program continuously. To get meaningful *debugging support* you may also specify the environment variable HALCONSPY (see chapter 2.5.2).
- *Result visualization*
HDevelop offers the immediate *inspection* of an operator's results. This visualization mechanism helps you to spot problems quickly. Additionally, it is very easy for you to modify graphics output parameters to optimize your graphics results.

2.9 Limitations

There are some HALCON limitations which you have to keep in mind, although you won't come even near these limits in most applications:

- Maximum image size: 32768×32768
- Maximum number of image matrices in memory¹³: 200
- Maximum number of objects per parameter: 100000
- Maximum number of channels per image: 500
- Maximum number of values in a tuple: 1000000
- Maximum number of contour points: 30000
- Maximum number of polygon control points: 10000
- Maximum value for coordinates: ± 32768
- Maximum length of strings: 1024 characters

¹³Not to be confused with image objects, since objects can share the underlying gray values.

Chapter 3

HDevelop Example Session

In this chapter you will learn how to solve an image analysis problem from scratch. During an example session the most important features of HDevelop are going to be introduced. You will become familiar with HDevelop's *graphical user interface (GUI)* and learn to use it efficiently.

The example is chosen from the quality control domain. The goal is to detect all *ball bondings* on a *die*, see Fig. 3.1. The HDevelop program corresponding to the example session is included in the HALCON package:

```
%HALCONROOT%\examples\hdevelop\Manuals\GettingStarted\example.dev
```

In particular it will be shown

- how to construct a HDevelop program with HALCON operators,
- how to find suitable operators and parameter values,
- how to handle iconic and control variables,
- how to handle the graphics window,
- how to generate a region of interest (ROI),
- how to solve the example problem using morphology,
- how to solve the example problem by pattern matching, and
- how to use control constructs.

Read this chapter very carefully and you will understand the basic interactive features of HDevelop. This will help you to develop your own sophisticated image analysis programs in the future.

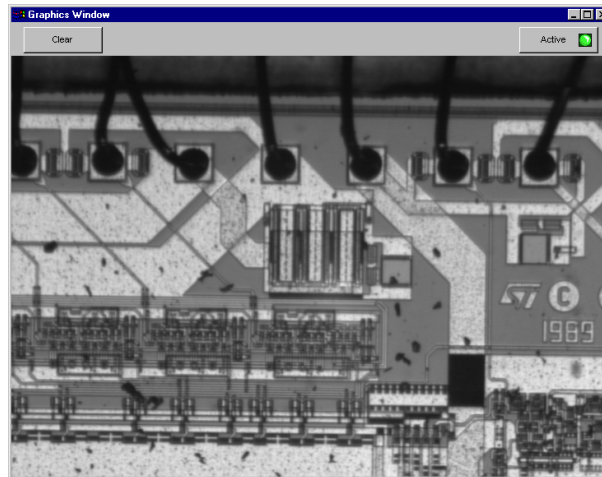


Figure 3.1: The image to be analyzed: All *ball bondings* on a die have to be detected.

3.1 Editing a HDevelop Program

After starting the interactive tool HDevelop, the GUI as described in chapter 2.6 appears. Now HDevelop waits for your input. To get your job done you have to create and to process variables (either iconic or control) by executing suitable HALCON or HDevelop operators. The program you develop during a HDevelop session is listed in the program window. To add a new line,¹ i.e., a new operator, you have to perform two steps:

- Position the *insertion cursor* on the appropriate line of the program where you would like to insert the new line. This cursor is placed by pressing the <Shift> button on your keyboard and clicking in the left column beside of the program line with the left mouse button.
- Select the new operator using the Operator menu or type its name in the *operator window's* text field.

Now the new operator is displayed in the *operator window*. Provide or adapt the input parameters and — if desired — change the names of the output variables. Then click Enter to insert the operator as a new program line. If you click on Ok instead, the operator is not only inserted, but also executed immediately. Finally, if you click on Apply the operator is not inserted into the program, but only executed. This allows you to test the effect of your parameter settings very quickly and conveniently.

To execute an arbitrary line of a HALCON program you have to position the *program counter (PC)* at this line (just click at the left column beside the line). Then press the Step icon in the main window tool bar. If you press Run instead all subsequent lines will be executed until a breakpoint or a stop command is encountered. A single-click on a program line selects this line, a double-click transfers the line back to the operator window.

¹The Copy, Cut, and Paste commands, which are also very convenient to edit a HDevelop program, will be explained later during this session.

Once you have finished the program you can save it and load it again later, or you might export it as C++ code to generate a standalone application (see File ▸ Save as...).

These basic steps will be used many times during the example session. So you will have many opportunities to get used to the different methods of editing program, specifying parameter values, etc. For now, just continue with applying your first operator to read an image.

3.2 Loading an Image

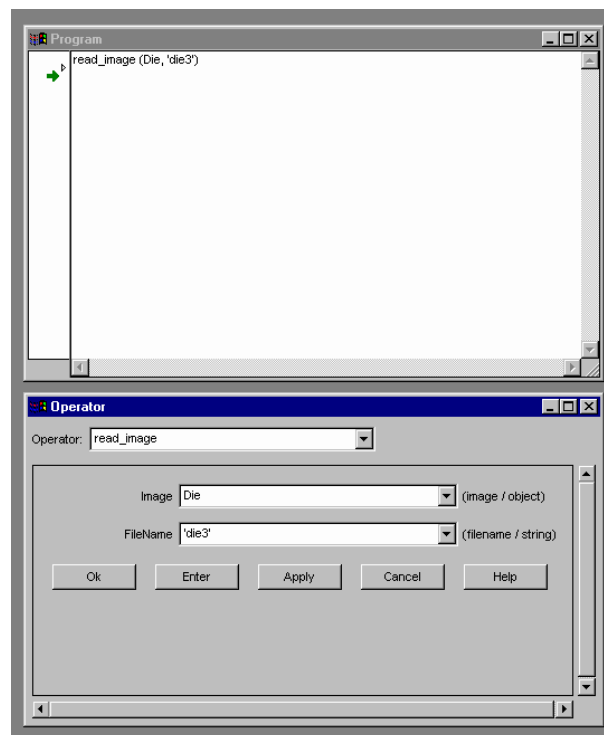


Figure 3.2: The parameterization of the `read_image` operator.

First of all, you have to create a HALCON object that contains your *input image*. You may obtain an image in several ways:

- You can use a framegrabber to obtain an online image (see menu item Operators ▸ Image ▸ Framegrabber).
- You can read an image from a file (see menu item Operators ▸ File ▸ Images and menu item File ▸ Read image...).
- You can create a new image (see menu item Operators ▸ Image ▸ Creation).

In the example we will use the `read_image` operator. Just select `read_image` in the submenu Operators ▸ File ▸ Images or type `read_image` in the *operator window's* text field and press Enter. Now the operator should be displayed in the *operator window*, see Fig. 3.2. The first parameter is the *image object* you are about to create. Change the default name of the

corresponding variable to 'Die'. You get a text field focus by clicking the left mouse button inside the desired text field. The second parameter is the name of the *image file*. Change it to 'die3':

```
read_image (Die,'die3')
```

Having specified all parameters you should press Ok to transfer the operator to the *program window* and to execute it. Now you have created the first line of your program in the *program window*. The variable Die containing the image object you just created is displayed in the *variable window*. Simultaneously, the result of the operator (in this case the original image) is displayed in the *graphics window*. If you encounter an error trying to load the image, check whether your environment is set up correctly. The image should be located in %HALCONROOT%\images which is also the default search path for image files as long as you do not provide specific search paths in the HALCONIMAGES variable, see section 2.5.

3.3 The Graphics Window

The size of the initial graphics window is 512×512 pixels. Thus if your image is bigger or smaller than this size it will be displayed distorted. You can interactively change the window size just by picking the lower right corner with the mouse, but the easiest way to see the undistorted image is to select Visualization ▸ Size Window ▸ Original. This enlarges or shrinks the window to the size of the most recently displayed image. However, in this example we are going to open a new window with the proper size because we will need its window handle later on. Again, there is a interactive way to do this (see menu Visualization), but we will use the HDevelop operator dev_open_window instead to include this task in the program itself. HDevelop operators are handled in the same way as HALCON operators. You can access them via the menu Operators ▸ HDevelop or directly via the *operator window's* text field. The parameter Width and Height specify the size of the window. If you do not know the size of the current image you just specify '-1' for both parameters. In that case the size of the most recently created image will be used.

```
dev_open_window (0,0,-1,-1,'black',WindowID)
```

Click on Ok and a new *graphics window* appears. It has the same size as your image. But the new graphics display remains dark. It is now the *active graphics window* and waits for your input. It is up to you what to display in this window. Double-click on the *iconic variable* Die in the *variable window* by using the left mouse button in order to display this image. Now you see the input image without distortion, see Fig. 3.1.

The default visualization for an image is to display the image matrix, i.e., the gray (or color) value of each pixel. You can easily switch to other visualization modes via Visualization ▸ Set parameters... ▸ paint. Choose for example '3D-plot' and select Update to see a 3D plot of the image. The various visualization modes are described in detail in the HDevelop Users Manual. Do not forget to change the mode back to 'default' before continuing with the example session. All the visualization modes can be modified within your program as well: For example, the display of the histogram is activated by dev_set_paint('3D-plot'). See also section 3.4 to learn how to manipulate the visualization of regions.

Furthermore, it is possible to zoom into the image. Just select the menu item Visualization ▸ Set parameters... ▸ zoom. Click on the upper Interactive... button. Now you can specify a rectangle within your graphics window. To do this you press the left mouse button to indicate the upper left corner. Keep the button pressed and drag the mouse to the lower right corner of the desired rectangle. Release the left button and click the right button once. Now you have specified the image part you are interested in. All visualizations are now zoomed to this image part. This is the interactive way to set an image part. Another possibility is the direct input of the coordinates of the image part (in Visualization ▸ Set parameters... ▸ zoom or to use the HDevelop operator `dev_set_part`). To visualize the whole image again, click Reset in Visualization ▸ Set parameters... ▸ zoom.

3.4 Generating a Region of Interest (ROI)

Very often it is useful to specify a ROI which is used in the following processing steps. By restricting the image domain the performance can be increased significantly because all following image processing tasks will be performed only within this ROI and not on the whole image.

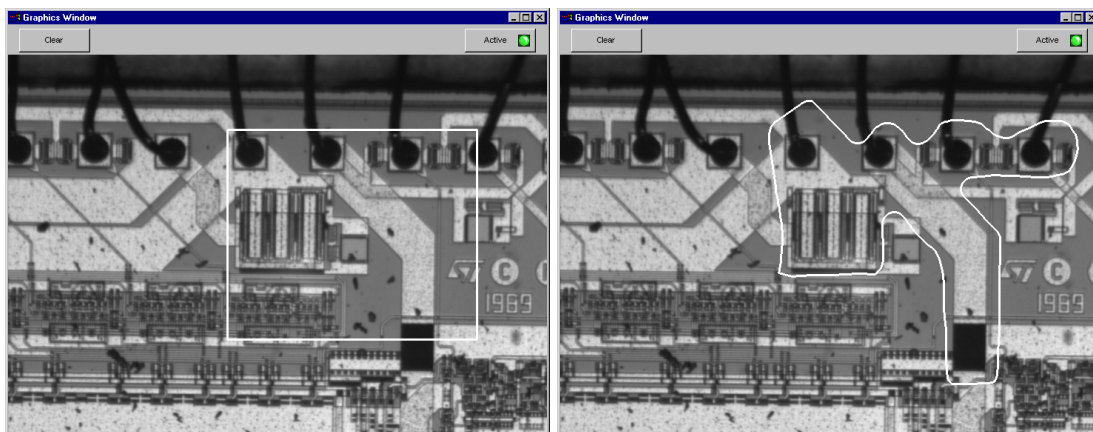


Figure 3.3: A rectangular and an arbitrary region of interest (ROI).

To generate a ROI you have the following possibilities:

- Interactive creation of a rectangular ROI:
Select the operator `draw_rectangle1` from the menu Operators ▸ Graphics ▸ Drawing to draw a rectangle in the *graphics window*. Just execute the operator with the default names for the output variables (containing the pixel coordinates of the upper left and lower right corner of the rectangle). This creates a new line in the *program window* and HDevelop waits for you to draw the rectangle (like you did before while setting an image part to zoom in). After you finished the drawing, the four control output variables Row1, Column1, Row2, and Column2 will appear in the *variable window*, see Fig. 3.4. To generate the corresponding ROI as an HALCON object, use the operator `gen_rectangle1` from the menu Operator ▸ Regions ▸ Creation with the variables Row1 – Column2 as input parameters. You might use 'ROI' as the name for the resulting output object:

```
draw_rectangle1 (WindowID, Row1, Column1, Row2, Column2)
gen_rectangle1  (ROI, Row1, Column1, Row2, Column2)
```

By executing this operator you create another type of HALCON object — a region, see section 4.2.3.

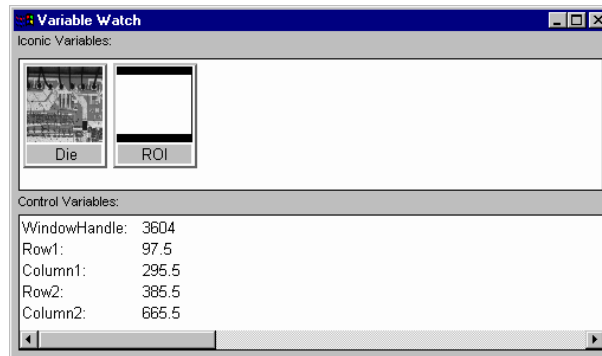


Figure 3.4: The variable window containing iconic and control variables.

The default visualization for a region is the shape of the region (here the rectangle) drawn with the current color into the *graphics window*. You may manipulate this graphics output by choosing items of the menu *Visualization* or executing the corresponding HDevelop operator `dev_*`. For example, set the drawing color to 'green' with *Visualization* ▸ *Color*, increase the line width for drawing to 3 pixels with *Visualization* ▸ *Line Width* and specify, that only the region boundary ('margin') instead of the whole region ('fill') should be displayed with *Visualization* ▸ *Draw*. Each time you change the visualization, the last iconic object that was drawn into the currently active graphics window is redisplayed (Fig. 3.3). Therefore, you need only redisplay an object by hand (by double-clicking it in the *variables window* if you want to change the display mode of an "older" object.

- **Interactive creation of an arbitrary ROI:**
HALCON supports arbitrary regions as ROI, see Fig. 3.3. Just use `draw_region` from the menu *Operators* ▸ *Graphics* ▸ *Drawing* instead of the sequence `draw_rectangle1` and `gen_rectangle1`:

```
draw_region (ROI, WindowID)
```

To adapt the program in this way, you may select the line `draw_rectangle1` (just click on that line) and deactivate it with *Edit* ▸ *Deactivate*. Do the same with the line `gen_rectangle1`.

- **Computation of a suitable ROI:**
The computation of a suitable ROI based on intermediate results of the image analysis is the most challenging but also the most flexible approach. In our example we will detect the board itself and use this as a ROI for the ball detection, see section 3.7.

Once you have generated a ROI you can reduce the image domain to this region with the operator `reduce_domain` from *Operators* ▸ *Image* ▸ *Domain*. Specify `Die` as input image and `ROI` as input region:

```
reduce_domain (Die, ROI, ImageReduced)
mean_image (ImageReduced, ImageMean, 11, 11)
```

The resulting image `ImageReduced` is only defined within the ROI. Clear the window and double-click on the variable `ImageReduced` to see this effect. You might apply a couple of operators to the original image and the reduced image to see the increase in performance. For example, try `mean_image` from Operators ▸ Filter ▸ Smoothing with `Die` as input image. Then clear the window and double-click on the `mean_image` program line to change the input image to `ImageReduced`.

To repeat the above sequence with different ROIs just place the PC on `draw_rectangle` or `draw_region` (depending on which method you have activated) and press Run. In order to stop the execution of the program before a specific operator is applied, place a *break point (BP)* in the program line containing this operator (Control key and mouse click in the left column).

3.5 Finding the Right Operator

You have created your first HALCON object — the original image. But how to continue? There are more than 800 operators to choose from. Naturally, the selection of a suitable operator sequence depends on the image analysis task you have to solve. In general, it is up to your experience and image analysis knowledge to make the right selection. However, HDevelop supports you in many ways during this program development.

The HALCON operators are hierarchically structured. Just browse the Operators menu to get a first impression.

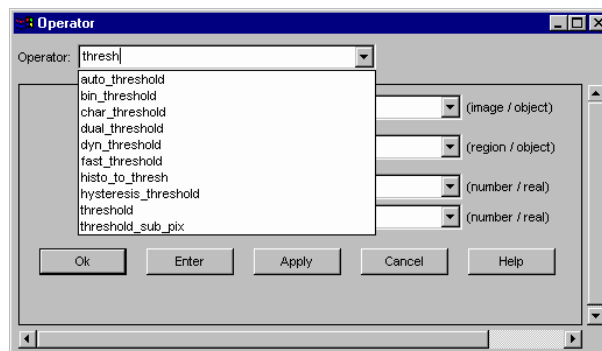


Figure 3.5: Selecting the threshold operator using the substring “thresh”.

Again, if you already know at least a substring of the operator name you might also just type this string in the *operator window*’s text field. If there is an ambiguity, you choose the operator inside the *combo box*, see Fig. 3.5.

For each HALCON operator the HDevelop *online help* provides an HTML description. Just select Help ▸ HTML... to start your default browser. You will see the table of contents of all image analysis modules — structured in the same way as the Operators menu. Simply select an appropriate chapter and continue browsing until you find your operator. There are also many cross references according to the relationships between the operators. At the end of the main

page there is an index of all operators, which helps you to jump directly to a specific operator HTML page.

Furthermore, the menu **Suggestions ▸ Keywords** offers access to HALCON operators by *keywords*. By selecting this menu item, a window is displayed with the available keywords on the left side and the operators related to the selected keywords on the right side. If you select a keyword using your mouse you will see the corresponding operators in the right area. Each mouse click in the left area marks the keyword and adds some operators in the right area. To finally select an operator and display it in the *operator window* just click on the operator name using your left mouse button.

Once you have selected one of the operators, like `read_image` in our example, you can enquire information related to this operator from the *operator knowledge base*:

- **The HTML online help.**
Just click on the **Help** button to launch your default web browser and get the complete online documentation of the selected operator.
- **Suggestions ▸ Alternatives**
If an intermediate result does not fulfill your requirements you might want to choose an alternative operator which might be more suitable for your task. Note, that in many situations there is a trade off between performance and accuracy of the results. For example, you might use the high performance smoothing filter `mean_image` in one situation and the higher quality smoothing filters `gauss_image` or `smooth_image` in another.
- **Suggestions ▸ See also**
The online help of HDevelop also offers references to other operators, which are not exactly alternatives, but related to the current operator in some way. For example, if you currently look at `read_image` you might be interested in the `write_image` operator as well.
- **Suggestions ▸ Predecessor**
Many operators require a specific input provided by other operators. For example, before computing junction points in a skeleton (`junctions_skeleton`), you have to compute the skeleton itself (`skeleton`).
- **Suggestions ▸ Successor**
On the other hand, there are typical sequences of operators. Hence HDevelop suggests some reasonable operators to be executed following the operator you are currently looking at. For example, thresholding might be a good idea after applying an edge filter.

In our example you have just created the input image. Naturally, this is a very unspecific situation with lots of different opportunities to continue. However, we will accept one of the successor suggestions, the `threshold` operator later on.

3.6 Finding the Right Parameter Values

Once you have selected an operator (from the menu, by typing its name, or by double-clicking on a program line in the program window), it is displayed in the *operator window*. Now you

have to specify the parameters for the operator. Of course, you can type in every parameter manually (either a control value or a variable name). In many cases it is more convenient to use the *combo boxes* (click on the arrow at the right side of the parameter field) instead, which provide a listing of meaningful values or names of variables you specified within your program. Furthermore, for control input parameters default values are provided by HDevelop.

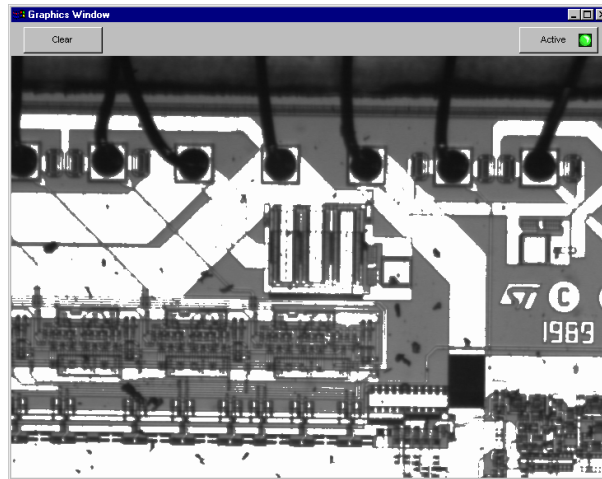


Figure 3.6: The region object BrightRegion containing all pixels with gray values larger or equal 128.

For example, apply a *threshold operator* to the input image by selecting the operator threshold from Operators ▸ Segmentation. Specify the input image Die using the *combo box*. If you use the default values for the lower and upper threshold, all pixels in the input image with *gray values* larger than 128 will be included in the resulting output region. Therefore, you could call the corresponding output variable BrightRegion. It might be helpful to set the visualization mode Visualization ▸ Draw to 'fill'. In this case you see all the selected image pixels painted in the current drawing color in the *graphics window*. Double-click on Die and BrightRegion again to underlay the visualization of the region with the original image, see Fig. 3.6.

To choose another parameter value just double-click on the program line containing the corresponding operator, change the parameter, and press Apply. The operator will be executed immediately and its result will be visualized. Repeat this process until you are satisfied with the result, then press Ok. But remember to clear the window or underlay the original image before each new execution of an operator returning a region. Otherwise, you only see the accumulated visualization of all regions you generate during the parameter tuning.

3.7 Finding the Board as ROI

Looking at the input image Die you will notice that the surface of the board is quite bright. Thus, by applying the threshold operator with the default values 128 and 255, a reasonable *raw segmentation* of the board can be achieved. We will now transform this *raw segmentation* into a *region of interest* for the ball detection: Select the HALCON operator shape_tans with BrightRegion (the result of the thresholding) as input region and 'rectangle2' as value for the parameter 'type' to obtain the *smallest rectangle* (of arbitrary orientation) containing BrightRegion. The resulting region (called 'ROI') is a reasonable approximation of the board

in the image. As in section 3.4, reduce the domain of the original image to this ROI with the `reduce_domain` operator:

```
threshold (Die, BrightRegion, 128, 255)
shape_trans (BrightRegion, ROI, 'rectangle2')
reduce_domain (Die, ROI, DieROI)
```

The following computations will be performed only on the resulting image `DieROI` within the board.

3.8 Finding Bonding Balls Using Morphology

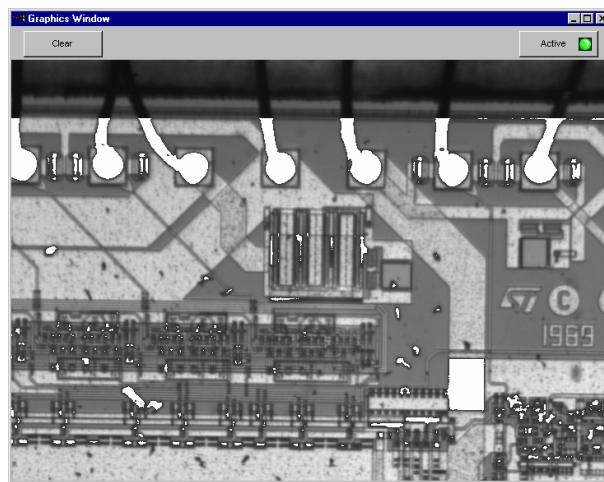


Figure 3.7: The region object `RawSegmentation` containing all dark pixels within the board.

If you look closer at your image you will notice that the bonding balls to detect are darker than the *die* itself. Therefore, another *thresholding* might be suitable as the next operation, see Fig. 3.7. Due to noise, the resulting region (all the dark parts within the board) might have some small holes which can obstruct further processing. For a controlled filling up of these artifacts, the HALCON operator `fill_up_shape` (in Operators > Regions > Transformation) is suitable.

```
threshold (DieROI, RawSegmentation, 0, 50)
fill_up_shape (RawSegmentation, Wires, 'area', 1, 100)
```

Visualize the variables `Wires` and `RawSegmentation`, then change the drawing color to see the difference.

Unfortunately, the region found so far contains not only the bonding balls but also some darker areas, especially the wires connected to the balls. Obviously, the bonding balls have a circular shape. We use this information to separate the bonding balls from the other dark parts by using the HALCON operator `opening_circle` (in Operators > Morphology > Region). This morphological operation will suppress all parts of the region, where the underlying structural element — a circle with a specific radius — does not fit within the region, see Fig. 3.8.

```
opening_circle (Wires, BallRegion, 15.5)
```

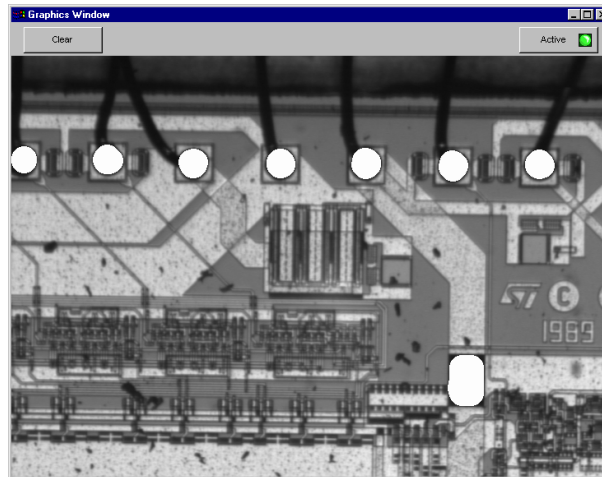


Figure 3.8: Removing faulty parts of the raw segmentation with the `opening_circle` operation.

Visualize the variables `Wires` and `BallRegion`, then change the drawing color to see the effect of this operation.

Now there is only one artifact left in the segmentation: A large dark rectangular shape, in which the circular mask for the opening has fitted as well. In order to eliminate this error the intermediate result — the region `BallRegion` — is separated into connected regions via the HALCON operator `connection`:

```
connection (BallRegion, Balls)
```

Note the difference: `BallRegion` is one region containing all the pixels of the segmentation so far. The different parts of this region are not connected, but still they form a single region. `Balls` is a variable containing seven separated HALCON objects — one per connected region. Set `Visualization > Colored` to 12. Now 12 different colors are used cyclically to paint the region objects. It is a very convenient feature of the HALCON system that iconic variables can contain not only single objects, but also tuples of objects. Most HALCON operators accept such tuples as input and perform their task simultaneously on all the contained objects. Due to this mechanism you won't have to program loops to process objects one by one in most situations.

Based on the circular shape of balls, the elimination of the rectangular region is now very simple. Just apply the `select_shape` operator (in `Operators > Regions > Features`) with 'circularity' as the selection criterion:

```
select_shape (Balls, FinalBalls, 'circularity', 'and', 0.85, 1.0)
```

Double-click on the variable `Die`, then on `FinalBalls` to see the selected regions only. Note again that `Balls` contains seven objects, from which you select all circular regions with this single operator call. Now, the ball segmentation is finished. In the next section it will be shown how to perform measurements based on the segmentation result. But before continuing, recall how simple it was to detect the balls and handle all the intermediate iconic objects generated:

```
read_image (Die, 'die3')
threshold (Die, BrightRegion, 128, 255)
shape_trans (BrightRegion, ROI, 'rectangle2')
reduce_domain (Die, ROI, DieROI)
```

```

threshold (DieROI, RawSegmentation, 0, 50)
fill_up_shape (RawSegmentation, Wires, 'area', 1, 100)
opening_circle (Wires, BallRegion, 15.5)
connection (BallRegion, Balls)
select_shape (Balls, FinalBalls, 'circularity', 'and', 0.85, 1.0)

```

3.9 Working with Control Variables

So far we have detected the ball bondings within the image. Based on this segmentation we can now perform some simple measurements. The most obvious measurement is to count the balls via `count_obj` (in Operators ▸ Object ▸ Information):

```
count_obj (FinalBalls, NumBalls)
```

Similar to the `draw_rectangle1` command in section 3.4 you generated a new control variable called `NumBalls` containing the desired number of balls. There are many different HALCON operators computing specific features of regions (see Operators ▸ Regions ▸ Features). Among them, one you might try now is `smallest_circle`. This operator determines the position and radius of the smallest circle containing each input region:

```
smallest_circle (FinalBalls, Row, Column, Radius)
```

Note that `FinalBalls` contains six objects. Thus, six circles are calculated — one for each of the input regions. Therefore, the output control variables `Row`, `Column`, and `Radius` do not contain single values, but tuples of values. Again, no loop construct is necessary to process all objects. Tuples of `ncontrol` values are shown in brackets within the variable window. If a tuple variable contains too many elements to be shown at once, the value list is truncated and three dots are added. For inspecting such a tuple, just double-click on the variable you want to see with the left mouse button. A window will be opened containing all the elements of the tuple.

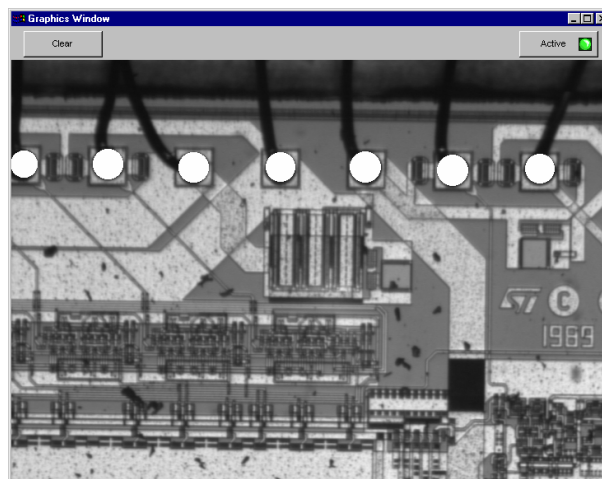


Figure 3.9: The final result of the ball detection based on morphology.

The further processing of control data is facilitated by a built-in formula interpreter. See the HDevelop User's Manual for details. For example, you could compute the average value of the tuple `Radius` with the following HDevelop assign statement (see Operator ▸ Control)

```
assign (meanRadius, sum(Radius) / |Radius|)
```

which is displayed in the program window as

```
meanRadius := sum(Radius) / |Radius|
```

A more reasonable measurement concerning the quality control task might be the computation of the minimum area of the balls. This can be achieved by applying the HALCON operator `area_center` (in Operators ▸ Regions ▸ Features), which calculates the area of each ball, followed by another assign statement:

```
area_center (FinalBalls, Area, RowCenter, ColCenter)
mimArea := min(Area)
```

Of course, tuples of control values can also be used as input parameters. For example, draw the circles you have computed with the HALCON operator `disp_circle` (in Operators ▸ Graphics ▸ Output), see Fig. 3.9:

```
disp_circle (WindowID, Row, Column, Radius)
```

3.10 Finding Bonding Balls Using Pattern Matching

The morphologic approach detected the desired bonding balls by suppressing spurious parts of the raw segmentation gained by thresholding. In this section you will encounter a completely different approach to this detection problem. The pattern matching approach can be divided into two steps:

1. Creating a template (typically during a training phase).
2. Searching for this template in the current image.

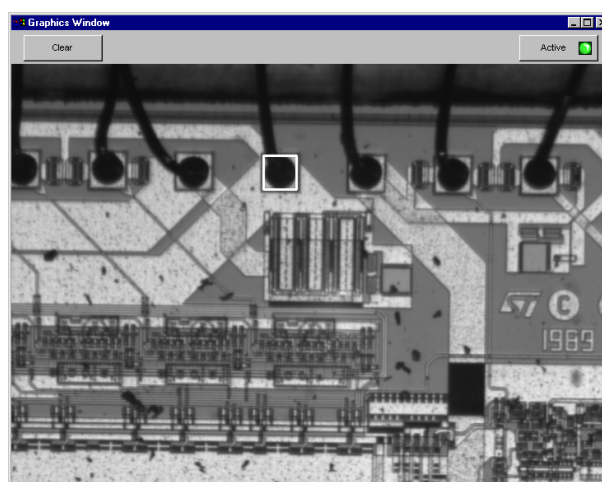


Figure 3.10: A template for a ball.

In our example we will create a template containing a single ball. To do this, just copy the two program lines you used to generate a rectangular search space (section 3.4):

```
draw_rectangle1 (WindowID, Row1, Column1, Row2, Column2)
gen_rectangle1  (ROI, Row1, Column1, Row2, Column2)
```

To do so, select the two program lines, copy them with `Edit ▷ Copy`, position the insertion cursor at the end of the program and select `Edit ▷ Paste`. Then, position the PC on the `draw_rectangle1` command, press `Run` and draw a small rectangle containing nothing but one of the balls, see Fig. 3.10.

Then copy the program line for reducing the image domain to the resulting region ROI:

```
reduce_domain (Die, ROI, ImageReduced)
```

Now you can transform `ImageReduced`, which is the original image restricted to the rectangular domain you just have specified, to a template. This done by the HALCON operator `create_template` (in `Operators ▷ Filter ▷ Match`):

```
create_template (ImageReduced, 5, 4, 'sort', 'original', TemplateID)
```

To obtain the center of the template (which will be needed for visualization purposes later) use the following assign statements:

```
TemplRow := (Row1 + Row2)/2
TemplCol := (Col1 + Col2)/2
```

After the creation of the template you have the choice between different matching algorithms. We will use a two step strategy. First, a fast matching operator is applied to find all image points where the template might match. In the second step, within each cluster of such points the exact position of the template will be determined. The HALCON operator `fast_match` (again in `Operator ▷ Filter ▷ Match`) delivers the region `Matches` containing all possible matching positions:

```
fast_match (Die, Matches, TemplateID, 20)
```

Double-click on the original image `Die` and the result region `Match` to see these positions. The input parameter `MaxError` is set to 20 in this example. There is a trade-off between performance and fault tolerance of the matching. Decrease the value and watch the execution time when applying the operator. For very small values of the tolerated error some of the balls might be missed. On the other hand, increasing the value decreases the performance significantly.

Besides `fast_match` there is also a HALCON operator `fast_match_mg` working on different resolution levels, which in general increases the performance. Just try

```
fast_match_mg (Die, Matches, TemplateID, 20, 3)
```

to perform a search with three resolution levels and compare the execution time with the one resolution method underlying `fast_match`.

After we have derived all possible matching positions, we decompose the corresponding region into connected components:

```
connection (Matches, BallROI)
```

If you specified a reasonable template and a suitable `MaxError` this should result in six regions corresponding to the six (fully) visible balls in the image.

```
count_obj (BallROI, NumBalls)
```

Within each of the regions in BallROI we would like to determine the exact position of the template. This can be done by another HALCON operator, called `best_match`.² BallROI contains only regions. Thus, we have to add an image, that is gray values, to the objects prior to the final matching. This is done with the operator `add_channels` (in Operators > Image > Channel):

```
add_channels (BallROI, Die, BallImage)
best_match (BallImage, TemplateID, 20, 'true', Row, Column, Error)
```

Note that the `best_match` operator is applied to all objects contained in BallImage. Thus, the control output variables Row and Column specifying the exact matching position and Error containing the matching error should be tuples of values. To visualize the results we display rectangles like the one used for the template generation — shifted by the offset between the matching positions and the center of the template, see Fig. 3.11.

```
disp_rectangle1 (WindowID, Row1+Row-TemplRow, Column1+Column-TemplCol,
                Row2+Row-TemplRow, Column2+Column-TemplCol)
```

You should make sure that the region visualization is set to 'margin' using Visualization > draw before executing this command.

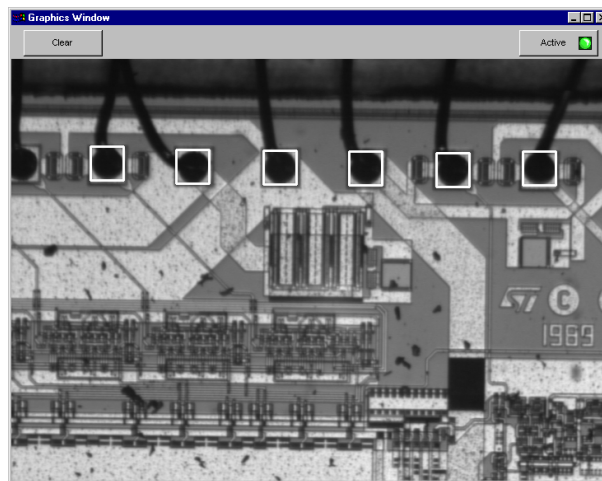


Figure 3.11: The final result of the ball detection based on pattern matching.

If you do not need the template any longer, it should be cleared with the HALCON operator `clear_template`. Summarizing, the ball detection using pattern matching was done by the following sequence of HALCON operators:

```
read_image (Die, 'die3')
draw_rectangle1 (WindowID, Row1, Column1, Row2, Column2)
gen_rectangle1 (ROI, Row1, Column1, Row2, Column2)
reduce_domain (Die, ROI, ImageReduced)
create_template (ImageReduced, 5, 4, 'sort', 'original', TemplateID)
```

²Again, there is a multi grid version of this matching operator, but since we already restricted the search to a very small part of the image, it is not too interesting to use `best_match_mg` in this case.

```

fast_match_mg (Die, Matches, TemplateID, 20, 3)
connection (Matches, BallROI)
add_channels (BallROI, Die, BallImage)
best_match (BallImage, TemplateID, 20, 'true', Row, Column, Error)
clear_template (TemplateID)

```

Note that the check for correct shape and area of each ball is implicitly done by the pattern matching. In the previous approach this was performed by the `opening_circle` operator.

By the way: Program lines that are not needed any more can be permanently removed. Just select them (left mouse button) and choose `Edit ▸ Delete`. Don't be shy — there is an `Edit ▸ Undo` as well. To temporarily deactivate a line use `Edit ▸ Deactivate` and `Edit ▸ Activate` to activate the line again.

3.11 Using Control Constructs

HDevelop offers various control statements to construct more complicated programs. For a detailed specification see the HDevelop Users Manual. You already have encountered the `assign` statement in the previous sections. We will now use two other control statements — `for` and `if` — to further improve the example program.

Up to now, all matching positions returned by `best_match` in section 3.10 were used to display a rectangle. This might lead to undesirable results if the matching fails for some of the input objects. Just decrease `MaxError` to 5, for example, to see what happens: For each input object in `BallImage` which does not contain the template 0 is returned in `Row` and `Column`³ and the maximum average gray value deviation of 255 is returned in `Error`.

The correct way to handle these matching errors is to check for all resulting matching positions whether they are valid or not. Simultaneously, the real number of detected balls can be computed: Initialize a counter with the `assign` statement

```
RealNumBalls := 0
```

Then program a loop with the counter `i` and the end value `NumBalls` (control statement `for`). Inside the loop, check the error value of each match and skip faulty matches:

```

for i := 1 to NumBalls by 1
  if (Error[i-1] < 255)
    disp_rectangle1 (Row1+Row[i-1]-TemplRow, Column1+Column[i-1]-TemplCol,
                    Row2+Row[i-1]-TemplRow, Column2+Column[i-1]-TemplCol)
    RealNumBalls := RealNumBalls + 1
  endif
endfor

```

Display the original image `Die` again and execute the above loop to see the result.

³A template position which is actually outside of the current image, because `Row` and `Column` specify the position of the *center* of the template.

Chapter 4

Philosophy of HALCON

HALCON is more than just a library of image processing operators or another graphical user interface for image processing. It is a comprehensive system for machine vision applications suitable for product development, research, and education. It also includes a powerful *computer aided vision engineering* tool called HDevelop. It might be helpful to have a closer look at the philosophy of HALCON to find out more about the different tools and features.

4.1 Facts about HALCON

Let's start with some facts describing the main characteristics of HALCON:

□ *A comprehensive operator data base supports efficient, flexible image processing and eases the development of image analysis programs.*

The operator data base consists of two parts — a library of image processing operators and a knowledge base. The library contains more than 900 operators. The operators can be used in any combination. All HALCON applications (e.g., HDevelop or HALCON C++ programs) make use of the library to perform image processing tasks. The operators exhibit a broad range of functionality from simple tasks (e.g., “read image from file”) to complex processes (e.g., “estimate state by Kalman filtering”). Basically they all have a rather lean complexity and are meant to process single steps of a task instead of the whole task. So, the typical solution of an image analysis problem consists of a combination of several HALCON operators. This concept allows much greater flexibility than others based on fewer, but more complex operators that are specialized to perform a certain task. Specialized operators are only suitable for special tasks and become worthless when the task changes. In contrast to this, HALCON operators may be used in any combination. If the task changes, the choice of the used HALCON operator set may change, too. It is just like the difference between a predefined automaton and a programmable computer. The first one may be easier to use, but is restricted to a few tasks. The second one is more complex to use, but can be adapted to changing problems and may perform nearly any task.

Among the great number of operators there are some that implement the same task by different algorithms. This allows to vary how fast and precise a task is performed. Consider pattern

matching, for example: For a task, where only the rough positions of matching points are needed, but where the results must be returned as fast as possible, the operator `fast_match` may be used. In contrast to this, `best_match` is the better choice when needing the exact positions (even with subpixel accuracy) by using a more time consuming algorithm.

Apart from the library, the *operator data base* provides a *knowledge base* that contains detailed information about all operators, such as their computational complexity, a plain text description of their functionality, or typical successor-/predecessor-operators. This operator description is used for the *automatic* generation of different language interfaces like C or C++, different manuals, such as the HTML and PDF versions of the reference manual, and the online help system. Furthermore, HDevelop uses the operator description to support the development of image analysis programs, e.g., by suggesting appropriate successors of operators. More information on the content of the knowledge base can be found in the HALCON C-Interface Programmers Guide in the chapter “Operator Description.”

□ *HALCON provides a tool HDevelop for interactive development of image analysis programs.*

In most cases, developing image analysis programs is a complex and time consuming task. To help the user find the right operators and parameters for a given task HALCON provides the *computer aided vision engineering tool* HDevelop. This eases developing programs in several ways: first, HDevelop has a *graphical user interface* that allows an easy, intuitive handling of operators and image data. Second, operators can be processed and combined interactively. All results are immediately visualized so that the user can experiment with different operators and parameter values and directly sees the result. Moreover, HDevelop supports the user by suggesting appropriate operators and parameter values. An *online help* explains how the operators work and illustrates their usage by examples. If you are satisfied with the result you can save the developed program either in a special HDevelop file format to load it again into HDevelop later on or export the program as C++ source code. By exporting, the program can be used by other image analysis applications and can be edited and optimized just like any “hand-written” image processing program.

□ *HALCON may easily be extended by new operators.*

Although HALCON already contains more than 900 operators for various tasks, you may wish to implement a new one, e.g., in order to access a special hardware or an operator using an alternative algorithm. To do so, HALCON provides the C-Interface that allows the integration of new operators (implemented in C). It contains several predefined routines and macros for easy handling of image data and memory objects in C. Once a new operator has been successfully integrated, it can be used like any other HALCON operator. The C-Interface Programmers Guide contains detailed information about extending the operator library.

□ *HALCON allows continued application of already implemented image processing software.*

Software engineering is a time consuming process and in most cases expensive as well. So, everyone is interested in using already implemented software as long as possible, and no one is bent on re-implementing their software with every installation of new software components. Therefore, HALCON supports the continued application of already implemented software: Convert your programs into new HALCON operators to use them within HALCON, which can be done via the C-Interface as described above. Moreover, HALCON hardly restricts the hardware configuration. A wide range of machines and systems can be used. Various framegrabbers

are supported. New ones may be easily integrated (see below). Thus you need not change your hardware configuration and therefore also need not to change your software.

□ *HALCON supports a wide range of platforms.*

HALCON is largely architecture independent and therefore allows you to use the system of your choice. It supports different platforms, such as Solaris, Tru64 UNIX, SGI-Irix, Linux, or Windows NT.

□ *New framegrabbers may easily be integrated.*

Although HALCON already supports several framegrabbers (e.g., from Imagenation, Bitflow, or Integral Technologies), you may wish to install and use a different one. To do so, HALCON provides a transparent framegrabber interface with sample source code for integrating new framegrabbers. The only thing you have to do is to adapt this code to a specific framegrabber and to link it to the HALCON code. The Framegrabber Integration Programmers Manual describes in detail how to do this.

□ *HALCON supports data exchange with other applications.*

For easy exchange of image data HALCON supports several file formats, such as Tiff, Sun-Raster, Gif, PNM, PCX, XWD, BMP, binary. Other data (e.g., integer result values for image coordinates) may be exchanged via the HALCON language interface.

□ *HALCON processes color and multichannel images.*

HALCON works on all kinds of image data — binary, monochrome, color, or multichannel images. You may use all these kinds of images in the same manner. Multichannel images are useful to work on images that were acquired by a multisensor system.

□ *HALCON allows to focus image processing on a region of interest.*

Each image object has a domain — its area of definition — that can be changed by the user. When performing an image operator, it is processed only within the domain of the input image. This concept allows to focus image processing and therefore to speed it up.

□ *HALCON provides an easy handling of tuple objects and processes tuple elements simultaneously.*

Tuples are very helpful in many areas of application, since they enable you to work with sets of images, regions, or control data. With HALCON, tuples of objects or control values can be handled just as one object. Operators can either work on a single or on a tuple object. You need not bother about the amount of elements within a tuple. You simply pass one variable containing the tuple to an operator. HALCON decomposes it and works simultaneously on the tuple elements.

□ *HALCON implements efficient, fast region processing.*

This speeds up the processing and eases the handling of regions. Regions may overlap and can be of any size. They are stored by an optimized encoding to reduce memory costs.

□ *HALCON provides very fast pattern matching operators.*

Pattern matching is very helpful in numerous areas of application, but often not used due to its costs of computation time. HALCON allows to use pattern matching where you want to, as it uses a very efficient implementation of various pattern matching algorithms.

□ *HALCON is based on an efficient management of image and data objects.*

The HALCON memory manager uses fast, efficient algorithms to speed up image processing. It provides mechanisms for controlled memory accesses and services for debugging, such as checking with every deallocation, whether any write access exceeded the allocated memory block. The HALCON image data base provides several services for object handling to allow a fast and transparent access to image data. Images are held in memory. To reduce memory costs data is shared by different objects whenever possible. The next section gives a more detailed description of the object hierarchy and the handling of image data as well as of control data.

4.2 Handling Iconic and Control Data

HALCON distinguishes two types of data: *iconic* and *control* data. *Iconic* data is related to the original image (images, regions, XLD objects; see below), whereas *control* data are all kinds of “numerical” values, such as integers, floating-point numbers, or strings. Both — *iconic* and *control* data — are processed according to the “tuple scheme” within HALCON.

4.2.1 Tuple Processing

Whenever an input or output parameter of a HALCON operator contains data, it may be a single object/value or a tuple of objects/values. If a tuple is passed HALCON processes the operator simultaneously on all tuple members. For example, if you want to process a median filter on six different images, you might call `median_image` six times with varying input images. Or you might generate a tuple containing all six images and call `median_image` once with the tuple as input object. HALCON filters all tuple elements simultaneously and returns a tuple containing six filtered images. We have seen another example before in chapter 3, where several regions have been extracted from an input image. The regions became elements of one region tuple. If you want to know the center of all regions, you simply have to pass this tuple to the operator `area_center`. HALCON then returns a tuple of integer pairs containing the pixel positions of all centers.

Now that we know how tuples are processed we can have a closer look at the different object classes of iconic data: images, regions, and XLD objects.

4.2.2 Image Objects

Image objects contain the pixel data for image processing. They may be tuple objects containing more than one image object or a single image object. Every single image object consists of one domain describing its area of definition and one or more channels containing the gray values of the pixels (cf. Fig. 4.1). The number of channels is not restricted. A monochrome image may only have one channel, a color image may contain three channels according to the RGB scheme, a multisensor image may have several channels. The domain of an image object may be of any size and is represented by a *region*. Thus it can have holes or may consist of several, unconnected areas (see Fig. 4.2). The default domain of an image object is the

smallest rectangle enclosing the image. It may be changed to any size, e.g., via the operator `reduce_domain`, so that every image has its individual domain. Introducing an area of definition for an image has the following advantage: all image operators work only within this image domain. This allows to focus the processing on a region of interest. The amount of data to work on becomes smaller so that the processing is speeded up. An example that explicitly makes use of this can be found as HDevelop program named `autobahn.dev` under the path `%HALCONROOT%\examples\hdevelop\Applications\Sequences`.

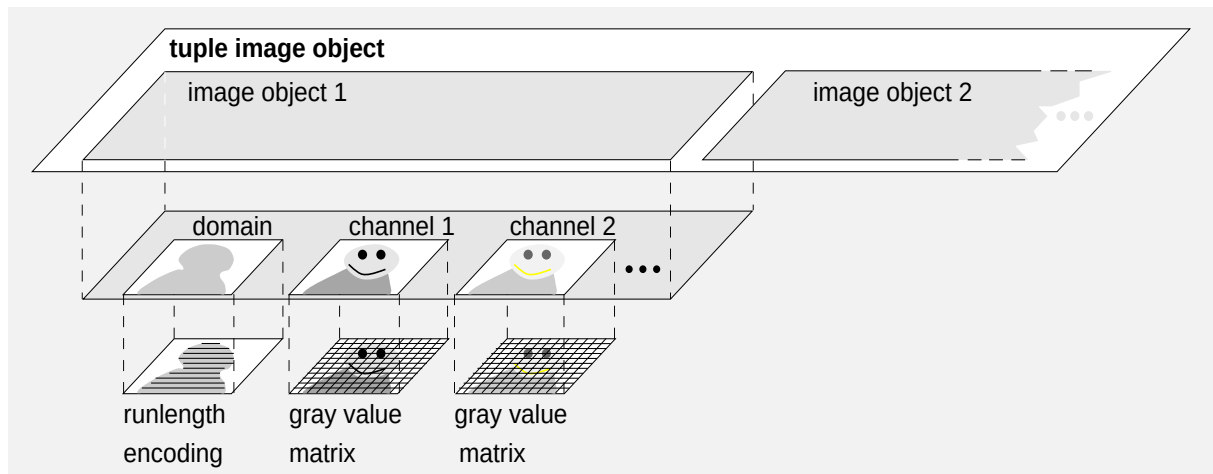


Figure 4.1: Structure of HALCON image objects.

4.2.3 Region Objects

Regions define areas in images, e.g., the domain of an image or the light areas of an image. Region objects consist of one or more regions. They are handled as separate objects independent from any image object, although they may represent an area within a specific image. This allows to use flexible shapes and sizes for regions and to store them efficiently. Only the region area must be represented in memory — independent from the gray values of the pixels. Moreover, this makes possible to define overlapping regions for the same image without the implicit merging that would happen when storing regions as matrices.

Region data is represented by a special variant of the runlength encoding — by a chord encoding : for every line (chord) of a region its row index (y-coordinate of chord; “line number”) and the x-coordinates of its start- and end-point is stored. Both — the start-point and the end-point — belong to the region. This allows a very compact representation of regions in memory and an efficient region processing, especially with morphologic operations.

Regions may be of any size and therefore even may exceed the image size. This is an additional advantage, since it prevents artifacts. For example, imagine you want to close a region with a circular structuring element to smooth the region’s boundaries and to close holes within the region, e.g., via the operator `closing_circle`. You use this to extract a little fin (see Fig. 4.3 or cf. the HDevelop program example `fin2.dev` under the path `%HALCONROOT%\examples\hdevelop\Applications\FA`). The closing is done by a dilation followed by an erosion, both with the same circular structuring element — in our case a circle

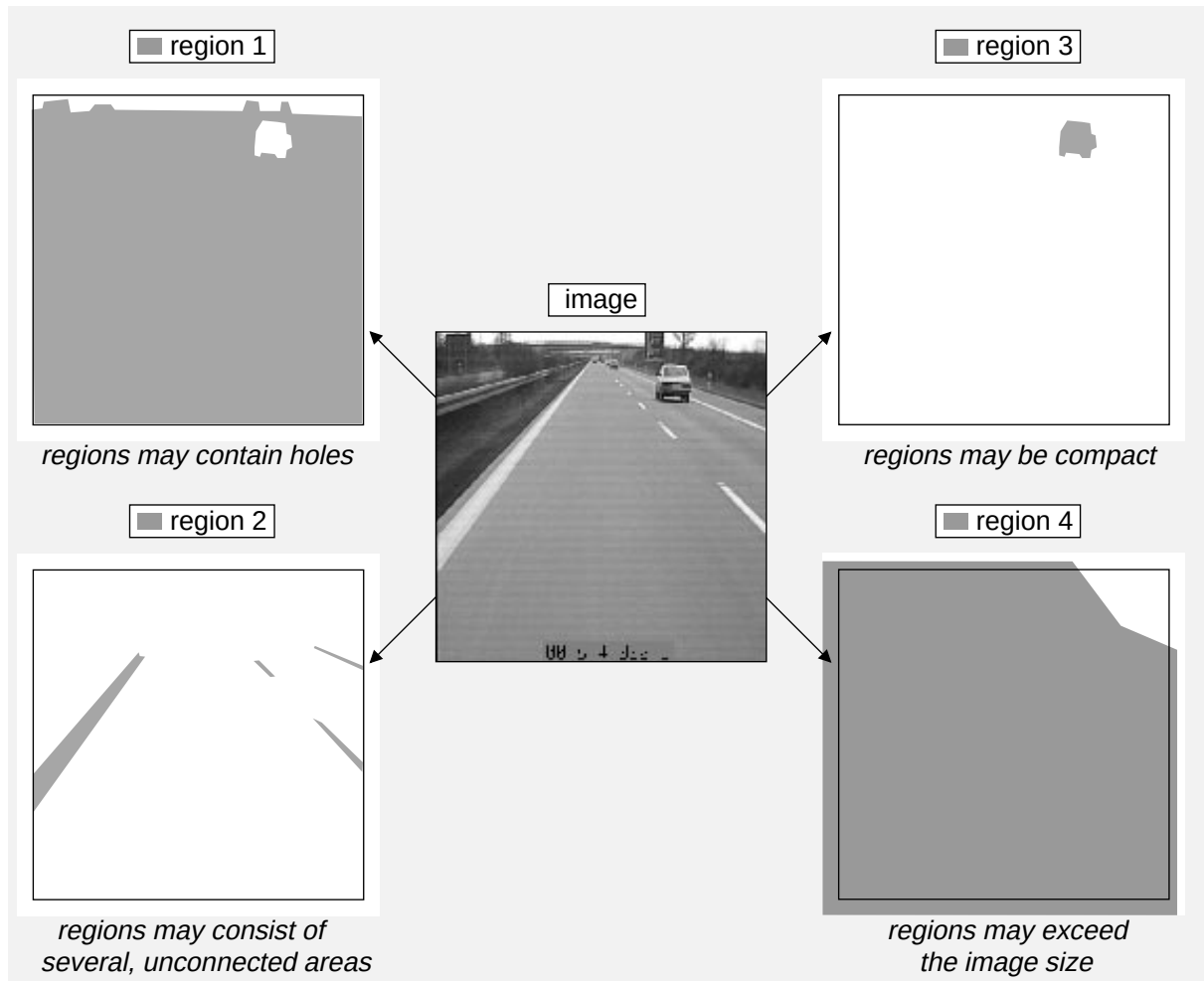


Figure 4.2: Different forms and sizes of regions.

of radius 250 (you may think that 250 is rather large, but with the advanced performance of HALCON this causes no performance problem). Now imagine that the maximum region size would be restricted to the image size. Thus after processing the first step (the dilation) the resulting region has to be clipped to the image size (see Fig. 4.3.a). The following erosion would work on a restricted area and therefore result in a much smaller area than the correct processing (see Fig. 4.3.b). This would make the correct extraction of the fin impossible.

4.2.4 XLD Objects

XLD objects (XLD = eXtended Line Description) describe areas or any closed or open contour. In contrast to regions, which represent all areas at pixel precision, XLD objects provide subpixel precision. There are two basic XLD structures: contours and polygons. See the HALCON C-Interface Programmers Guide for internal details. XLD objects can also contain additional information about points and lines, such as the gradient strength or the local orientation.

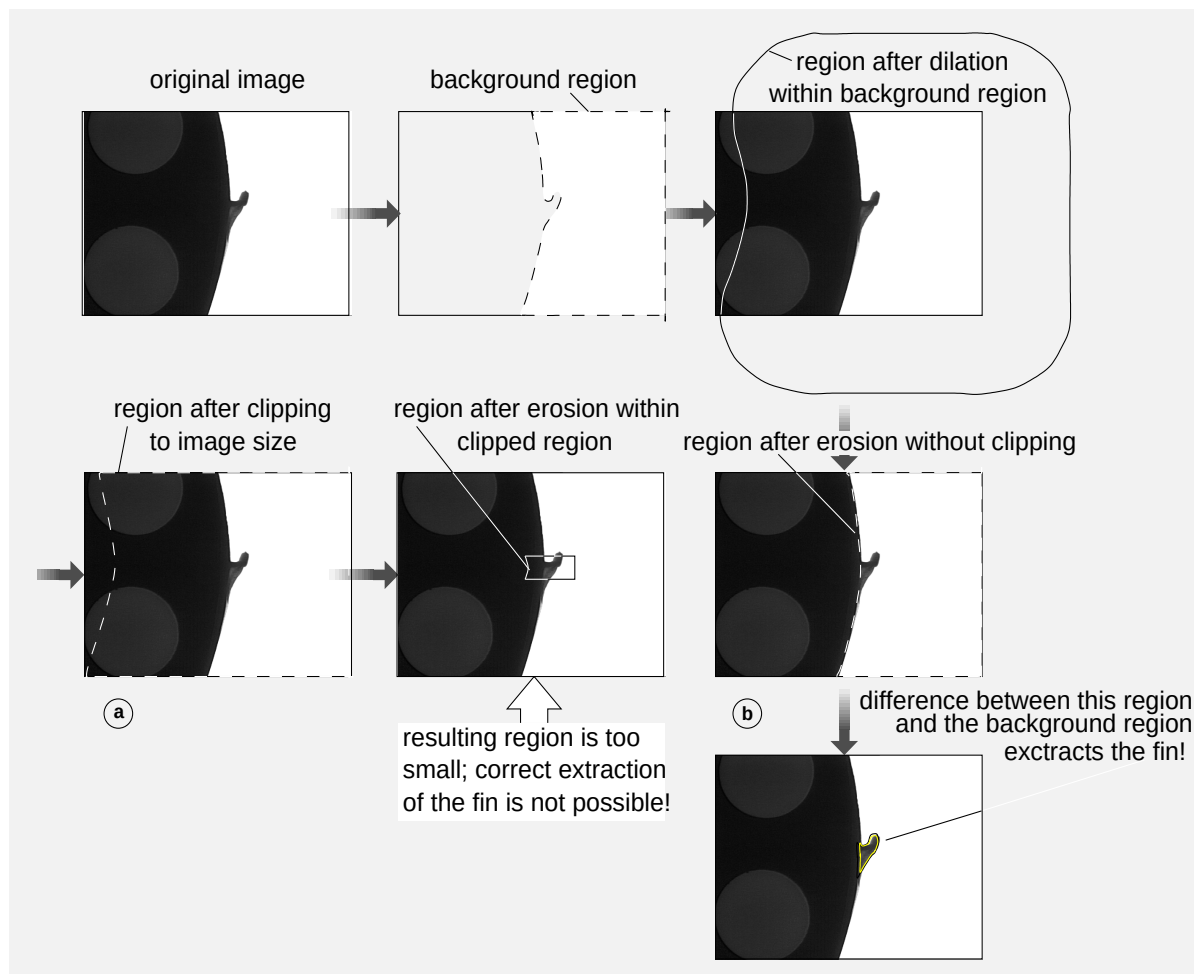


Figure 4.3: Example of region processing.

Bibliography

- [BB82] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [Jai92] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall International, Inc., 1992.
- [Rus92] J.C. Russ. *The Image Processing Handbook*. CRC Press, Boca Raton, Florida, 1992.

List of Figures

2.1	HALCON architecture	6
3.1	Ball bondings on a die.	22
3.2	The read_image operator.	23
3.3	A rectangular region of interest.	25
3.4	The variable window containing iconic and control variables.	26
3.5	Selecting an operator by a substring of it's name.	27
3.6	Detecting bright image parts by thresholding.	29
3.7	Detecting dark image parts inside the board.	30
3.8	Removing faulty parts of the raw segmentation with an opening operation. . . .	31
3.9	The final result of the ball detection based on morphology.	32
3.10	A template for a ball.	33
3.11	The final result of the ball detection based on pattern matching.	35
4.1	Structure of HALCON image objects.	41
4.2	Different forms and sizes of regions.	42
4.3	Example of region processing.	43

Index

- Aerial image interpretation, 1
- Architecture, 6
- Area of definition, 39–41
- area_center, 40
- Ball bondings, 21
- best_match, 38
- BP, 27
- Break point, 17, 27
- C, 5, 38
- C++, 5, 38
 - Export of source code, 38
- C-Interface, 38
- CAVE, 1
- Chord encoding, 41
- closing_circle, 41
- Color images, 39
- COM, 5
- Combo box, 17, 27
- Computer Aided Vision Engineering, 1
- Contour, 42
- Control data, 40
- Cross references, 19
- Data
 - Control, 40
 - Exchange, 39
 - Floating point numbers, 40
 - Iconic, 40
 - Image, 40
 - Integers, 40
 - Pixel, 40
 - Strings, 40
- Debugging, 5, 17, 19, 40
- Dilation, 41
- Domain, 40, 41
- Environment variables
 - LD_LIBRARY_PATH, 14
- Environment variables, 13
- ARCHITECTURE, 14
- DISPLAY, 15
- HALCONEXTENSIONS, 15
- HALCONIMAGES, 14
- HALCONROOT, 13
 - doc/html/reference/hdevelop, 14
 - help, 14
 - images, 14
 - license, 14
 - lut, 14
 - ocr, 14
- HALCONSPY, 14
- HOME, 15
- Erosion, 41
- Example session, 21
- Extended line description, 42
- FA, 1
- Factory automation, 1
- fast_match, 38
- Framegrabber, 39
- Graphical user interface, 15
- Graphics window, 15, 17, 24, 29
- Gray value, 29, 40
- Halcon, 5
 - Installation, 6
 - Library, 13
 - Operator, 22
- Halcon configuration, 13
 - Unix, 13
 - Windows NT, 13
- HDevelop, 13, 38
 - Program, 22
- Help, 18
- HORUS, 6
- Iconic data, 40
- Iconic variable, 24
- Image, 40

- Color, 39
- Matrix, 40
- Multichannel, 39
- Object, 40
- Image file, 24
- Image file formats, 39
 - Binary, 39
 - BMP, 39
 - Gif, 39
 - PCX, 39
 - PNM, 39
 - Sun-Raster, 39
 - Tiff, 39
 - XWD, 39
- Image object, 23
- Insertion cursor, 17, 22
- Installation, 6, 8
- Keywords, 28
- Knowledge base, 37
- Language interface, 38, 39
- License
 - Development, 12
 - Dongle, 12
 - Floating, 7, 8, 11
 - Key, 12
 - Nodelocked, 7, 11
 - Runtime, 7, 12
- Limitations, 19
 - Maximum image size, 19
 - Maximum length of strings, 19
 - Maximum number of channels per image, 19
 - Maximum number of contour points, 19
 - Maximum number of image matrices in memory, 19
 - Maximum number of objects per parameter, 19
 - Maximum number of polygon control points, 19
 - Maximum value for coordinates, 19
- Loading an image, 23
- Main window, 15
 - Menu, 16
 - Edit, 16
 - Execute, 16
 - File, 16
 - Help, 16
 - Operators, 16
 - Suggestions, 16
 - Visualization, 16
 - Window, 16
- Tool bar, 16
 - Activate, 16
 - Copy, 16
 - Cut, 16
 - Deactivate, 16
 - Gray histogram info, 16
 - New, 16
 - Open, 16
 - Paste, 16
 - Pixel info, 16
 - Region info, 16
 - Reset, 16
 - Run, 16
 - Save, 16
 - Set parameters, 16
 - Step, 16
 - Stop, 16
 - Zooming, 16
- median_image, 40
- Medical image analysis, 1
- Memory management, 40
- Multichannel images, 39
- Object
 - Image, 40
 - Region, 41
 - Tuple, 39, 40
 - XLD, 42
- Online help, 19, 27
- Operating systems, 39
- Operator data base, 37
- Operator knowledge base, 6, 19, 28
- Operator library, 37
 - Extension, 38
- Operator window, 15, 17, 22, 23
 - Text field, 22–24, 27
- Packages, 15
- Pattern matching, 39
- PC, 22
- Performance, 28
- Platforms, 39

- Linux, 39
- SGI-Irix, 39
- Solaris, 39
- Tru64 UNIX, 39
- Windows NT, 39
- Portable applications, 5
- Program counter, 17, 22
- Program creation, 16
- Program window, 15, 17, 24
- Quality control, 1
- Raw segmentation, 29
- Re-using code, 38
- reduce_domain, 41
- Region, 41
 - Form, 42
 - Object, 41
 - Size, 42
- Region of interest, 25, 29, 39–41
- Remote sensing, 1
- Result visualization, 19
- ROI, 25, 39, 40
- Runlength encoding, 41
- Session management, 16
- Subpixel precision, 42
- Surveillance tasks, 1
- System suggestions, 19
 - Alternatives, 19
 - Keywords, 19
 - Predecessors, 19
 - References, 19
 - Successors, 19
- Troubleshooting, 17
- Tuple, 39, 40
- Tuple object, 39, 40
- Uninstallation, 8
- Unix, 5, 11–13
- User extensions, 15
- Variable window, 15, 17, 24
- Visual Basic, 5
- Visualization
 - Images, 24
 - Regions, 26, 29
- Windows 2000, 7
- Windows NT, 5, 7, 12, 13
- XLD, 42